# A Multiple Shooting Descent-based Filter Method for Optimal Control Problems

Gisela C. V. Ramadas, Edite M. G. P. Fernandes, Ana Maria A. C. Rocha and M. Fernanda P. Costa

**Abstract** A direct multiple shooting (MS) method is implemented to solve optimal control problems (OCP) in the *Mayer form*. The use of an MS method gives rise to the so-called 'continuity conditions' that must be satisfied together with general algebraic equality and inequality constraints. The resulting finite nonlinear optimization problem is solved by a first-order descent method based on the filter methodology. In the equivalent tri-objective problem, the descent method aims to minimize the objective function, the violation of the 'continuity conditions' and the violation of the algebraic constraints simultaneously. The numerical experiments carried out with different types of benchmark OCP are encouraging.

Gisela C. V. Ramadas
Research Center of Mechanical Engineering (CIDEM)
School of Engineering of Porto (ISEP), Polytechnic of Porto, 4200-072 Porto, Portugal
e-mail: gcv@isep.ipp.pt

Edite M. G. P. Fernandes
ALGORITMI Center
e-mail: emgpf@dps.uminho.pt

Ana Maria A. C. Rocha
ALGORITMI Center,
Department of Production and Systems,
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
e-mail: arocha@dps.uminho.pt

M. Fernanda P. Costa
Centre of Mathematics,
Department of Mathematics,
University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
e-mail: mfc@math.uminho.pt

# 1 Introduction

An optimal control problem (OCP) is a constrained optimization problem that has a set of dynamic equations as constraints. Application domains of OCP are varied [1]. There are three types of OCP that differ in the formulation of the functional to be optimized. For example, an OCP of the *Lagrange form* has the objective functional in its pure integral form as shown

$$\begin{aligned} J^* = \min_{\mathbf{u}(t)\in U} \; & J(\mathbf{y}(t),\mathbf{u}(t)) \equiv \int_0^T f_2(t,\mathbf{y}(t),\mathbf{u}(t))\,dt \\ \text{s.t. } & \mathbf{y}'(t) = \mathbf{f}_1(t,\mathbf{y}(t),\mathbf{u}(t)), \;\; \text{for } t \in [0,T] \\ & \mathbf{y}(0) = \mathbf{y}_0,\; \mathbf{y}(T) = \mathbf{y}_T\;, \end{aligned} \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^{\bar{s}}$ is the vector of state variables of the dynamic system, $\mathbf{u} \in U \subset \mathbb{R}^c$ is the vector of control or input variables and $U$ represents a class of functions (in particular functions of class $C^1$ and piecewise constant) and usually contains limitations to the control [2]. To convert problem (1) into the *Mayer form*, a new variable is added to the states vector $\mathbf{y}$, such that $y_s'(t) = f_2(t,\mathbf{y}(t),\mathbf{u}(t))$ with the initial condition $y_s(0) = 0$, where $s = \bar{s}+1$ represents the total number of state variables. Thus, problem (1) becomes:

$$\begin{aligned} \min_{\mathbf{u}(t)\in U} \; & J(\mathbf{y}(t),\mathbf{u}(t)) \equiv y_s(T) \\ \text{s.t. } & \mathbf{y}'(t) = \mathbf{f}_1(t,\mathbf{y}(t),\mathbf{u}(t)) \\ & y_s'(t) = f_2(t,\mathbf{y}(t),\mathbf{u}(t)), \;\; \text{for } t \in [0,T] \\ & \mathbf{y}(0) = \mathbf{y}_0,\; y_s(0) = 0,\; \mathbf{y}(T) = \mathbf{y}_T\;. \end{aligned} \tag{2}$$

In the OCP we want to find $\mathbf{u}$ that minimizes the objective functional $J$ subject to the dynamic system of ordinary differential equations (ODE). The problem may have other more complex 'terminal constraints' $H(T,\mathbf{y}(T),\mathbf{u}(T)) = 0$. States $\mathbf{y}$ and control $\mathbf{u}$ may also be constrained by algebraic equation constraints $h_e(t,\mathbf{y}(t),\mathbf{u}(t)) = 0$, $e \in E$ and 'path constraints' $g_j(t,\mathbf{y}(t),\mathbf{u}(t)) \le 0$, $j \in F$, where $E = \{1,2,\ldots,m\}$ and $F = \{1,2,\ldots,l\}$.

Methods for solving OCP like (2) can be classified into indirect and direct methods. Indirect methods use the first-order necessary conditions from Pontryagin's maximum principle to reformulate the original problem into a boundary value problem. On the other hand, direct methods solve the OCP directly [3] transforming the infinite-dimensional OCP into a finite-dimensional optimization problem that can be solved by effective and well-established nonlinear programming (NLP) algorithms. All direct methods discretize the control variables but differ in the way they treat the state variables [4] . They are also classified as *Discretize then Optimize* strategies in contrast to the *Optimize then Discretize* strategies of the indirect methods [1] .

This paper explores the use of a first-order descent method based on the filter methodology [5, 6] to solve the NLP problem, within a direct method for solving an OCP in the *Mayer form*. The use of a direct multiple shooting (MS) method gives rise to the so-called 'continuity conditions' that must be satisfied. The novelty here

is that a filter methodology is used to minimize the objective function, the violation of the 'continuity conditions' and the violation of algebraic constraints simultaneously. The NLP problem is a tri-objective problem and the first-order descent method generates a search direction that is either the negative gradient of one of the functions to be minimized or a convex combination of negative gradients of two functions. To overcome the drawback of computing first derivatives, the gradients are approximated by finite differences.

The paper is organized as follows. Section 2 briefly describes the direct MS algorithm for solving the OCP in the *Mayer form*. The herein proposed first-order descent filter algorithm is discussed in Sect. 3, the numerical experiments are shown in Sect. 4 and we conclude the paper with Sect. 5.

## 2 Direct Multiple Shooting Method

In a direct single shooting (SS) method, only the controls are discretized in the NLP problem [3] . The dynamic system is solved by an ODE solver to get the state values for the optimization. Thus, simulation and optimization are carried out sequentially. On a specific grid defined by $0 = t_1 < t_2 < \cdots < t_{N-1} < t_N = T$, where $N-1$ is the total number of subintervals, the control $\mathbf{u}(t)$ is discretized, namely using piecewise polynomial approximations. The simplest of all is a piecewise constant, $\mathbf{u}(t) = \mathbf{q}^i$, for $t \in [t_i, t_{i+1}]$ and $i = 1, \ldots, N-1$ so that $\mathbf{u}(t)$ only depends on the control parameters $\mathbf{q} = (\mathbf{q}^1, \mathbf{q}^2, \ldots, \mathbf{q}^{N-1})$ and $\mathbf{u}(t) = \mathbf{u}(t, \mathbf{q})$. When the horizon length $T$ is not fixed, the control parameter vector also includes $T$ to define the optimization variables. The dynamic system is solved by (forward numerical integration) an ODE solver and the state variables $\mathbf{y}(t)$ are considered as dependent variables $\mathbf{y}(t, \mathbf{q})$. The main advantage of a direct SS method is the reduced number of decision variables (control parameters) in the NLP even for very large dynamic systems. However, unstable systems may be difficult to handle.

In a direct MS method, discretized controls and state values at the start nodes of the grid (grid points) – $\mathbf{x}^i \in \mathbb{R}^s$, $i = 1, 2, \ldots, N-1$, known as MS node variables – are the decision variables for the NLP solver [7]. After the discretization of the controls, the ODE system is solved on each shooting subinterval $[t_i, t_{i+1}]$ independently, but they need to be linked by the auxiliary variables $\mathbf{x}^i$, $i = 1, 2, \ldots, N-1$. They are the initial values for the state variables for the $N-1$ independent initial value problems on the subintervals $[t_i, t_{i+1}]$:

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t), \mathbf{q}^i) \equiv \begin{cases} \mathbf{f}_1(t, \mathbf{y}(t), \mathbf{q}^i) \\ f_2(t, \mathbf{y}(t), \mathbf{q}^i) \end{cases} \text{ with } \mathbf{y}(t_i) = \mathbf{x}^i, \text{ for } t \in [t_i, t_{i+1}] \,,$$

where $\mathbf{y} \in \mathbb{R}^s$. Trajectories $\mathbf{y}^i(t; \mathbf{x}^i, \mathbf{q}^i)$ are obtained where the notation "$(t; \mathbf{x}^i, \mathbf{q}^i)$", for the argument, means that they are dependent on $t$ as well as on the specified values for the node variables $\mathbf{x}^i$ and control parameters. The initial state values $\mathbf{x}^i$ should satisfy the 'continuity conditions'

$$\mathbf{y}^i(t_{i+1};\mathbf{x}^i,\mathbf{q}^i) = \mathbf{x}^{i+1},\ i = 1,\ldots,N-1\ , \tag{3}$$

(ensuring continuity of the solution trajectory), the initial value $\mathbf{x}^1 = \mathbf{y}_0$ and the final state constraints $\mathbf{x}^N = \mathbf{y}_T$ [4, 8].

We choose to implement a direct MS method since it can cope with differential and algebraic equations that show unstable dynamical behavior [7]. The main steps of the direct MS algorithm are shown in Algorithm 1.

**Input:** $T$, $N$, $\mathbf{f}(t,\mathbf{y},\mathbf{u})$, $\mathbf{y}_0$, $\mathbf{y}_T$, constraint functions.
**Output:** Optimal control and state variables.
Define the grid points in the interval $[0,T]$: $0 = t_1 < \cdots < t_{N-1} < t_N = T$.
Discretize the control: $\mathbf{u}(t) = \mathbf{q}^i$ for $t \in [t_i,t_{i+1}]$, $i = 1,\ldots,N-1$.
Define the starting values for the state vector $\mathbf{x}^i$ for each $[t_i,t_{i+1}]$, $i = 1,\ldots,N-1$, and $\mathbf{x}^N$.
(Invoke the NLP algorithm)
**while** *Stopping conditions are not satisfied* **do**
  With $\mathbf{q}^i, i = 1,\ldots,N-1$, $\mathbf{x}^i, i = 1,\ldots,N$, use an ODE solver to evaluate the state trajectories in $[t_i,t_{i+1}]$, $i = 1,\ldots,N-1$:
  for $\mathbf{y}^i(t_i) = \mathbf{x}^i$, $(\mathbf{y}^i)'(t) = \mathbf{f}(t,\mathbf{y}^i(t),\mathbf{q}^i)$;
  Evaluate the 'continuity conditions' $\mathbf{y}^i(t_{i+1};\mathbf{x}^i,\mathbf{q}^i) = \mathbf{x}^{i+1}$, $i = 1,\ldots,N-1$, as well as $\mathbf{x}^1 = \mathbf{y}_0$ and $\mathbf{x}^N = \mathbf{y}_T$;
  Evaluate algebraic equality and inequality constraints for $t \in [t_i,t_{i+1}], i = 1,\ldots,N-1$;
  Evaluate the objective function;
  Generate new $\mathbf{q}^i, i = 1,\ldots,N-1$ and $\mathbf{x}^i, i = 1,\ldots,N$.
**end**

**Algorithm 1:** Direct MS algorithm

## 3 First-Order Descent Filter Method

The herein proposed first-order descent filter method relies on descent directions for two constraint violation functions (handled separately) and for the objective function in order to converge towards the optimal solution of the NLP problem. One of the constraint violation functions emerges from the 'continuity constraints' violation (including initial state and final state constraints) and the other comes up from the state and control algebraic equality and inequality constraints. We assume that the NLP problem is a non-convex constrained optimization problem (COP). For practical purposes, we assume that the OCP is in the *Mayer form*, the ODE system has initial and boundary state values, state and control variables are constrained by algebraic equality and inequality constraints, and the explicit 4th. order Runge-Kutta integration formula is used to solve the dynamic system in each subinterval $[t_i,t_{i+1}]$ using 5 points.

As stated in the last section, the decision variables of the COP are the initial state values at the nodes $\mathbf{x}^i \in \mathbb{R}^s$, $i = 1,\ldots,N$ and the control variables $\mathbf{q}^i \in \mathbb{R}^c$, $i = 1,\ldots,N-1$. Besides possible algebraic constraints on the state and control

variables, the 'continuity constraints' (3), the initial state and the final state constraints must be added to the optimization problem formulation. Thus, our COP has the following form:

$$\begin{aligned} \min_{\mathbf{x}^i, i\in I_N; \mathbf{q}^i, i\in I} \quad & y_s(T) \\ \text{s.t.} \quad & g_j(\mathbf{y}^i(t;\mathbf{x}^i,\mathbf{q}^i),\mathbf{q}^i) \le 0, t \in [t_i, t_{i+1}], i \in I, j \in F \\ & h_e(\mathbf{y}^i(t;\mathbf{x}^i,\mathbf{q}^i),\mathbf{q}^i) = 0, t \in [t_i, t_{i+1}], i \in I, e \in E \\ & \mathbf{y}^i(t_{i+1};\mathbf{x}^i,\mathbf{q}^i) - \mathbf{x}^{i+1} = 0, i \in I \\ & \mathbf{x}^1 - \mathbf{y}_0 = 0, \mathbf{x}^N - \mathbf{y}_T = 0\,, \end{aligned} \tag{4}$$

where $I = \{1,\ldots,N-1\}$ and $I_N = I \cup \{N\}$. To solve the optimization problem (4), the set of ODE must be solved so that the 'continuity constraints' $\mathbf{y}^i(t_{i+1};\mathbf{x}^i,\mathbf{q}^i) - \mathbf{x}^{i+1} = 0$, the initial state and the final state constraints, the other equality and inequality constraints and the objective function are evaluated (see Algorithm 1). Since problem (4) has constraints, we seek optimal values for $\mathbf{x}$ and $\mathbf{q}$ such that all the constraints are satisfied – a feasible solution of the COP – and the objective function takes the least value.

### 3.1 Filter Methodology

To check solution feasibility, a measure for the violation of the constraints is adopted. To implement the herein proposed filter methodology, the constraints are fractionated into two sets and their violations are computed and handled separately. We denote the violation of the 'continuity constraints', initial state and final state constraints by the non-negative function:

$$\theta(\mathbf{x},\mathbf{q}) = \sum_{l\in L}\sum_{i\in I}(y_l^i(t_{i+1};\mathbf{x}^i,\mathbf{q}^i) - x_l^{i+1})^2 + \sum_{l\in L}(x_l^1 - y_{l_0})^2 + \sum_{l\in L}(x_l^N - y_{l_T})^2\,, \tag{5}$$

where $L = \{1,2,\ldots,s\}$, noting that $\theta(\mathbf{x},\mathbf{q})$ is zero if the solution $(\mathbf{x},\mathbf{q})$ satisfies these constraints, and is positive otherwise. These are the constraints that are more difficult to be satisfied and we need to prioritly drive the violation $\theta$ to zero as soon as possible so that the ODE integration runs as close as possible to the exact values of the state variables.

To evaluate the algebraic equality and inequality constraints violation, a non-negative function $p$, also based on the Euclidean norm of vectors, is used

$$p(\mathbf{x},\mathbf{q}) = \sum_{j\in F}\sum_{i\in I}\max\left\{0, g_j(\mathbf{y}^i(t;\mathbf{x}^i,\mathbf{q}^i),\mathbf{q}^i)\right\}^2 + \sum_{e\in E}\sum_{i\in I} h_e(\mathbf{y}^i(t;\mathbf{x}^i,\mathbf{q}^i),\mathbf{q}^i)^2, \tag{6}$$

and similarly, $p(\mathbf{x},\mathbf{q}) = 0$ when the corresponding constraints are satisfied, and $p(\mathbf{x},\mathbf{q}) > 0$ otherwise. The violation of these constraints is also forced to converge to zero.

The extension of the filter methodology [5] into the descent algorithm to solve the COP is equivalent to the reformulation of the problem (4) as a tri-objective optimization problem that aims to minimize both the feasibility measures, defined by the constraint violation functions $\theta(\mathbf{x},\mathbf{q})$ and $p(\mathbf{x},\mathbf{q})$, and the optimality measure defined by the objective function $y_s(T)$:

$$\min_{\mathbf{x}^i, i\in I_N; \mathbf{q}^i, i\in I} (\theta(\mathbf{x},\mathbf{q}), p(\mathbf{x},\mathbf{q}), y_s(T)) . \tag{7}$$

In our filter methodology, a *filter* $\mathscr{F}$ is a finite set of triples $(\theta(\mathbf{x},\mathbf{q}), p(\mathbf{x},\mathbf{q}), y_s(T))$ that correspond to points $(\mathbf{x},\mathbf{q})$, none of which is dominated by any of the others in the *filter*. A point $(\hat{\mathbf{x}},\hat{\mathbf{q}})$ is said to dominate a point $(\mathbf{x},\mathbf{q})$ if and only if the following conditions are satisfied simultaneously:

$$\theta(\hat{\mathbf{x}},\hat{\mathbf{q}}) \leq \theta(\mathbf{x},\mathbf{q}),\ p(\hat{\mathbf{x}},\hat{\mathbf{q}}) \leq p(\mathbf{x},\mathbf{q}) \text{ and } \hat{y}_s(T) \leq y_s(T) ,$$

with at least one inequality being strict. The *filter* is initialized to $\mathscr{F} = \{(\theta, p, y_s) : \theta \geq \theta_{\max}, p \geq p_{\max}\}$, where $\theta_{\max}, p_{\max} > 0$ are upper bounds on the acceptable constraint violations. Let $\mathscr{F}_k$ be the *filter* at iteration $k$ of the algorithm. To avoid the acceptance of a trial point $(\bar{\mathbf{x}},\bar{\mathbf{q}})$ (approximation to the optimal solution), or the corresponding triple $(\theta(\bar{\mathbf{x}},\bar{\mathbf{q}}), p(\bar{\mathbf{x}},\bar{\mathbf{q}}), \bar{y}_s(T))$, that is arbitrary close to the boundary of the *filter*, the conditions of acceptability to the *filter* define an envelope around the filter and are as follows:

$$\begin{array}{l} \theta(\bar{\mathbf{x}},\bar{\mathbf{q}}) \leq (1-\gamma)\theta(\mathbf{x}^{(l)},\mathbf{q}^{(l)}) \text{ or } p(\bar{\mathbf{x}},\bar{\mathbf{q}}) < (1-\gamma)p(\mathbf{x}^{(l)},\mathbf{q}^{(l)}) \\ \text{or } \bar{y}_s(T) \leq y_s^{(l)}(T) - \gamma\Big(\theta(\mathbf{x}^{(l)},\mathbf{q}^{(l)}) + p(\mathbf{x}^{(l)},\mathbf{q}^{(l)})\Big) \end{array} \tag{8}$$

for all points $(\mathbf{x}^{(l)},\mathbf{q}^{(l)})$ that correspond to triples $(\theta(\mathbf{x}^{(l)},\mathbf{q}^{(l)}), p(\mathbf{x}^{(l)},\mathbf{q}^{(l)}), y_s^{(l)}(T))$ in the *filter* $\mathscr{F}_k$. Points with constraint violations that exceed $\theta_{\max}$ or $p_{\max}$ are not acceptable. The constant $\gamma \in (0,1)$ is fixed and the smaller the tighter is the envelope of acceptability. The above conditions impose a sufficient reduction on one of the feasibility measures or on the optimality measure for a point to be acceptable. When the point is acceptable to the *filter*, the *filter* is updated and whenever a point is added to the *filter*, all the dominated points are removed from it.

### 3.2 The First-Order Descent Filter Algorithm

The proposed first-order descent method is based on using gradient approximations of the functions, $\theta$, $p$ or $y_s$, of the tri-objective problem (7), to define search directions coupled with a simple line search to compute a step size that gives a simple decrease on one of the measures $\theta$, $p$ or $y_s$. Since $\theta$ is the most difficult to reduce, priority is given to searching along the (negative) gradient of $\theta$ or a (negative) combination of the gradient of $\theta$ with the gradient of $p$ or $y_s$. See Algorithm 2. For easy

of notation $\mathbf{v} = \left(x_1^1, \ldots, x_s^1, \ldots, x_1^N, \ldots, x_s^N, q_1^1, \ldots, q_c^1, \ldots, q_1^{N-1}, \ldots, q_c^{N-1}\right)^T$ is used to denote the vector of the decision variables ($\mathbf{v} \in \mathbb{R}^{n_D}$ , $n_D = Ns + (N-1)c$).

Each component $i$ of the gradient of $\theta$ with respect to the variable $v_i$, at an iteration $k$, is approximated by

$$\nabla_i \theta(\mathbf{v}^{(k)}) \approx \left(\theta(\mathbf{v}^{(k)} + \varepsilon \mathbf{e}_i) - \theta(\mathbf{v}^{(k)})\right)/\varepsilon \ , \ i = 1, 2, \ldots, n_D \tag{9}$$

for a positive and sufficiently small constant $\varepsilon$, being the vector $\mathbf{e}_i \in \mathbb{R}^{n_D}$ the $i$ column of the identity matrix. Similarly for the gradients approximation of $p$ and $y_s$.

To identify the best point computed so far, the below conditions (10) are imposed. Let $\mathbf{v}^{best}$ be the current best approximation to the optimal solution of problem (7). A trial point, $\bar{\mathbf{v}}$, will be the best point computed so far (replacing the current $\mathbf{v}^{best}$) if one of the conditions

$$\Theta(\bar{\mathbf{v}}) < \Theta(\mathbf{v}^{best}) \text{ or } \bar{y}_s(T) < y_s^{best}(T) \tag{10}$$

holds, where $\Theta = \theta + p$. At each iteration, the algorithm computes a trial point $\bar{\mathbf{v}}$, approximation to the optimal solution, by searching along a direction that is the negative gradient of $\theta$, or a negative convex combination of the gradients of $\theta$ and $p$, $\theta$ and $y_s$, or $p$ and $y_s$, at the current approximation $\mathbf{v}$. The selected direction depends on information related to the magnitude of $\theta$ and $p$, at $\mathbf{v}$. For example, if $p(\mathbf{v})$ is considered sufficiently small, i.e., $0 \leq p(\mathbf{v}) \leq \eta_1$, while $\theta(\mathbf{v}) > \eta_1$ (for a small error tolerance $\eta_1 > 0$), then the direction is the negative gradient of $\theta$ at $\mathbf{v}$. The search for a step size $\alpha \in (0, 1]$ goals the reduction of $\theta$ ('$M \leftarrow \theta$' in Algorithm 2). On the other hand, if both $p$ and $\theta$ are considered sufficiently small, then the direction is the negative convex combination of the gradients of $\theta$ and $y_s$, although the search for $\alpha$ forces the reduction on $\theta$.

If both $\theta$ and $p$ are not small yet (situation that occurs during the initial iterations) the direction is along the negative convex combination of the gradients of $\theta$ and $p$, although the line search forces the reduction on $\theta$. However, if $0 \leq \theta(\mathbf{v}) \leq \eta_1$ but $p(\mathbf{v}) > \eta_1$, then the direction is along the negative convex combination of the gradients of $p$ and $y_s$ and the line search forces the reduction on $p$. Further details are shown in the Algorithm 2.

The new trial point is accepted for further improvement if it satisfies the conditions to be acceptable to the current filter (see conditions (8)), although each trial point is considered as a new approximation to the optimal solution only if it is better than the previously saved best point, according to (10). In this situation, a new *outer* iteration - indexed by $k$ in Algorithm 2 - is carried out unless the convergence conditions are satisfied (see (11) below). If the trial point is accepted but it does not satisfy (10), $\theta$, $p$ and $y_s$ are evaluated at the trial point and a new *inner* iteration - indexed by $It$ - is carried out. This *inner* iterative process runs for a maximum of $It_{\max}$ iterations.

The trial point might not be acceptable to the *filter*, in which case another *inner* iteration is tried. If the number of iterations with non acceptable trial points reaches $It_{\max}$, the new direction is along the negative convex combination of the gradients of

**Input:** $N$, $T$, $k_{\max} > 0$, $It_{\max} > 0$, $\eta_1 > 0$
**Output:** $\mathbf{v}^{best}, \theta^{best}, p^{best}, y_s^{best}$
Set $k = 0$, exit = "false"; Initialize $\mathscr{F}$;
Set initial $\mathbf{v}$;
Compute $\theta = \theta(\mathbf{v}), p = p(\mathbf{v}), y_s = y_s(T)$; Update $\mathscr{F}$;
Set $\mathbf{v}^{best} = \mathbf{v}, \theta^{best} = \theta, p^{best} = p, y_s^{best} = y_s$;
**while** $k < k_{\max}$ *and exit = "false"* **do**
  Set $k = k+1$, $It = 0$, $it_{no} = 0$, accept = "true", stop = "false";
  **while** $It < It_{\max}$ *and stop = "false"* **do**
    Set $It = It + 1$, $F_{It} = It/It_{\max}$;
    Compute $\mathbf{G}_\theta \approx \nabla\theta(\mathbf{v})$, $\mathbf{G}_p \approx \nabla p(\mathbf{v})$, $\mathbf{G}_{y_s} \approx \nabla y_s(T)$ using (9);
    **if** *accept = "true"* **then**
      **if** $\theta \le \eta_1$ *and* $p \le \eta_1$ **then**
        Set $\mathbf{G} = (1 - F_{It})\mathbf{G}_\theta + F_{It}\mathbf{G}_{y_s}$; $M \leftarrow \theta$;
      **else**
        **if** $p \le \eta_1$ *and* $\theta > \eta_1$ **then**
          Set $\mathbf{G} = \mathbf{G}_\theta$; $M \leftarrow \theta$;
        **else**
          **if** $\theta \le \eta_1$ *and* $p > \eta_1$ **then**
            Set $\mathbf{G} = (1 - F_{It})\mathbf{G}_p + F_{It}\mathbf{G}_{y_s}$; $M \leftarrow p$;
          **else**
            Set $\mathbf{G} = (1 - F_{It})\mathbf{G}_\theta + F_{It}\mathbf{G}_p$; $M \leftarrow \theta$;
          **end**
        **end**
      **end**
    **else**
      Set $it_{no} = it_{no} + 1$;
      **if** $it_{no} < (It_{\max} - 1)$ **then**
        Set $\mathbf{G} = (1 - F_{It})\mathbf{G}_\theta + F_{It}\mathbf{G}_p$; $M \leftarrow \theta$;
      **else**
        Set $\mathbf{G} = (1 - F_{It})\mathbf{G}_{y_s} + F_{It}\mathbf{G}_\theta$; $M \leftarrow y_s$;
      **end**
    **end**
    Compute $\alpha \in (0, 1]$ such that $M(\mathbf{v} - \alpha\mathbf{G}) < M(\mathbf{v})$; Set
     $\bar{\mathbf{v}} = \mathbf{v} - \alpha\mathbf{G}, \bar{\theta} = \theta(\bar{\mathbf{v}}), \bar{p} = p(\bar{\mathbf{v}}), \bar{y}_s = \bar{y}_s(T)$;
    **if** $\bar{\mathbf{v}}$ *is acceptable to* filter *(according to* (8)*)* **then**
      Set $\mathbf{v} = \bar{\mathbf{v}}, \theta = \bar{\theta}, p = \bar{p}, y_s = \bar{y}_s$;
      Set accept = "true"; Update $\mathscr{F}$;
      **if** $\bar{\mathbf{v}}$ *is the best computed so far (see* (10)*)* **then**
        $\mathbf{v}^{best} = \bar{\mathbf{v}}, \theta^{best} = \bar{\theta}, p^{best} = \bar{p}, y_s^{best} = \bar{y}_s$;
        **if** *convergence conditions* (11) *are satisfied* **then**
          Set stop = "true", exit = "true" (convergence);
        **end**
        Set stop = "true";
      **end**
    **else**
      Set accept = "false";
    **end**
  **end**
**end**

**Algorithm 2:** Descent-filter algorithm

$\theta$ and $y_s$ (with a reduction on $y_s$ in the line search); otherwise, the negative convex combination of the gradients of $\theta$ and $p$ (with a reduction on $\theta$ in the line search) is tested.

The convergence conditions are said to be satisfied at a new trial point – the best point computed so far, $\mathbf{v}^{best}$, – if

$$\theta(\mathbf{v}^{best}) < \eta_1 \text{ and } p(\mathbf{v}^{best}) < \eta_1 \text{ and } perror = \left( \left| y_s^{best} - y_s^{pr.best} \right| / \left| y_s^{best} \right| \right) < \eta_2, \tag{11}$$

for small error tolerances $\eta_1 > 0$ and $\eta_2 > 0$, where the superscript *pr.best* refers to the previous best point. The *outer* iterative process also terminates if the number of iterations exceeds $k_{\max}$.

## 4 Numerical Experiments

The new direct MS method based on descent directions and the filter methodology has been tested with seven OCP. The MATLAB® (MATLAB is a registered trademark of the MathWorks, Inc.) programming language is used to code the algorithm and the tested problems. The numerical experiments were carried out on a PC Intel Core i7–7500U with 2.7GHz, 256Gb SSD and 16Gb of memory RAM. The values set to the parameters are shown in Table 1.

**Table 1** Parameter values

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $\theta_{\max}$ | $1E+03\,\theta(\mathbf{v}^{(0)})$ | $\eta_1$ | $1E-04$ |
| $p_{\max}$ | $1E+03\max\{p(\mathbf{v}^{(0)}),1\}$ | $\eta_2$ | $1E-03$ |
| $\gamma$ | $1E-05$ | $k_{\max}$ | 750 |
| $\varepsilon$ | $1E-06$ | $It_{\max}$ | $s$ |

First, three problems with free terminal time $T$ are solved. A simple approach is to apply the change of variable $t = T\tau$, (with $dt = Td\tau$) which transforms the problem into a fixed boundary problem on the interval $[0,1]$ and treats $T$ as an auxiliary variable. When the objective is to minimize $T$, an alternative is to add a new variable to the states vector $\mathbf{y} \in \mathbb{R}^{s-1}$ such that $y_s'(t) = 1$, with initial value $y_s(0) = 0$.

**Problem 1** A simple car model (*Dubins car*) is formulated with three degrees of freedom where the car is imagined as a rigid body that moves in a plane [2]. The position of the car is given by $(x, y, \beta)$ where $x$ and $y$ are the directions and $\beta$ is the angle with the $X$ axis. The problem is to drive in minimum time the car from a position to the origin:

$$\begin{aligned}
&\min_{u(t)} J(x(t),y(t),\beta(t),u(t)) \equiv T \\
&\text{s.t. } x'(t) = \cos(\beta(t)) \\
&\quad y'(t) = \sin(\beta(t)) \\
&\quad \beta'(t) = u(t),\ t \in [0,T] \\
&\quad x(0) = 4, y(0) = 0, \beta(0) = \tfrac{\pi}{2}, x(T) = 0, y(T) = 0, \\
&\quad |u(t)| \leq 2,\ t \in [0,T]\ .
\end{aligned}$$

The results from both strategies to handle $T$ free are shown in Table 2. The initial guesses were $x(t_i) = 2, y(t_i) = 0, \beta(t_i) = 1, i \in I_N$ and $u(t_i) = 0, i \in I$. The number of points considered in $[0,T]$ is 11. The table shows the values of $J$, $\theta$ and $p$ achieved at iteration $k$, as well as the number of function evaluations, $nfe$, and the time in seconds, $time$. Optimal solution reported [2] is $J^* = 4.32174$. The results are considered quite satisfactory. We show in Figs. 1(a) and 1(b) the optimal states trajectory and control respectively, obtained from the run that considers the change of variable $t \to \tau$. Fig. 1(c) displays the optimal control required to achieve identical states trajectory from the run that adds a new state variable. Slightly different optimal controls were obtained to reach identical states trajectory.

**Table 2** Results obtained for the problems 1, 2 and 3

| Problem | Handling $T$ | $k$ | $J$ | $\theta$ | $p$ | $nfe$ | $time$ |
|---|---|---|---|---|---|---|---|
| *Dubins car* | | 1 | 4.7539 | $2.7003E+01$ | $0.0000E+00$ | | |
| | adding new $y_s$ | 388 | 4.3329 | $9.9908E-05$ | $0.0000E+00$ | 44255 | 50.0 |
| | change of variable | 192 | 4.3658 | $9.6149E-05$ | $0.0000E+00$ | 18044 | 19.8 |
| *R allocation* | | 1 | 0.5714 | $1.0484E+02$ | $0.0000E+00$ | | |
| | adding new $y_s$ | 472 | 0.7219 | $9.7083E-05$ | $0.0000E+00$ | 44486 | 48.9 |
| | change of variable | 638 | 0.7232 | $9.9168E-05$ | $0.0000E+00$ | 47223 | 49.7 |
| *Zermelo* | | 1 | 3.8500 | $1.3863E+01$ | $0.0000E+00$ | | |
| | adding new $y_s$ | 323 | 3.5143 | $9.6343E-05$ | $2.8735E-05$ | 29595 | 32.3 |
| | change of variable | 644 | 3.5249 | $9.9618E-05$ | $9.6530E-06$ | 46160 | 48.2 |

**Problem 2** The resource allocation problem (*R allocation*) goals the assignment of resources in minimum time [2]:

$$\begin{aligned}
&\min_{u(t)} J(\mathbf{y}(t),\mathbf{u}(t)) \equiv T \\
&\text{s.t. } y_1'(t) = u_1(t)y_1(t)y_2(t) \\
&\quad y_2'(t) = u_2(t)y_1(t)y_2(t),\ t \in [0,T] \\
&\quad y_1(0) = 1, y_2(0) = 2, y_1(T)y_2(T) = 10, \\
&\quad y_1(t) \geq 0, y_2(t) \geq 0, u_1(t) + u_2(t) = 1, u_1(t) \geq 0, u_2(t) \geq 0,\ t \in [0,T]\ .
\end{aligned}$$

Since $u_2 = 1 - u_1$ the control vector can be reduced to a scalar $u_1 \equiv u \in [0,1]$. Using the initial guesses $y_1(t_i) = 1, y_2(t_i) = 0, i \in I_N$, $u(t_i) = 0, i \in I$ and $N = 11$, the results are shown in Table 2. Optimal solution reported [2] is $J^* = 0.714118$. Figures 1(d) and 1(e) show the optimal states $y_1, y_2$ and control $u_1, u_2$ respectively, for the case

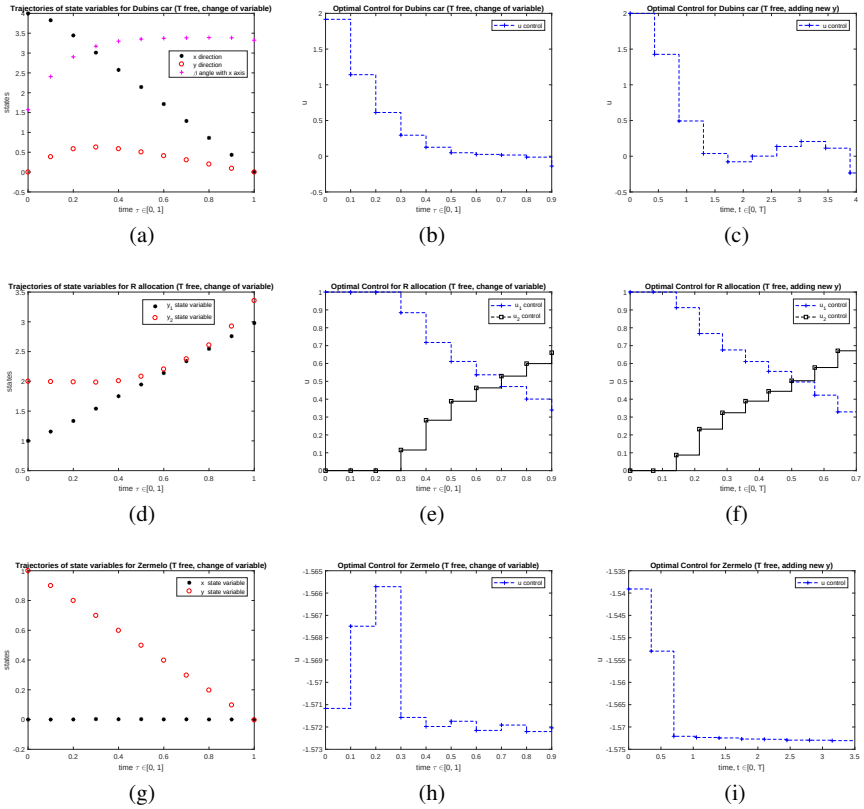


**Fig. 1** **a** States trajectory for *Dubins car*. **b** Optimal control for *Dubins car*. **c** Optimal control for *Dubins car* (when adding new $y_s$). **d** States trajectory for *R allocation*. **e** Optimal control for *R allocation*. **f** Optimal control for *R allocation* (when adding new $y_s$). **g** States trajectory for *Zermelo*. **h** Optimal control for *Zermelo*. **i** Optimal control for *Zermelo* (when adding new $y_s$)

where a change of variable is applied. Figure 1(f) shows the control for the case of handling $T$ free through the adding of a new state variable. The states trajectory are similar to Fig. 1(d).

**Problem 3** Consider an unmanned aerial vehicle (*Zermelo*) flying in a horizontal plane with constant speed $V$, although the heading angle $u(t)$ (control input) (with respect to the $X$ axis) can be varied. Winds are assumed to be in the $Y$ direction with speed $w$. The objective is to fly from point A=(0,1) to B=(0,0) in minimum time:

$$\begin{aligned}
&\min_{u(t)} J(x(t),y(t),u(t)) \equiv T \\
&\text{s.t. } x'(t) = V\cos(u(t)) \\
&\quad y'(t) = V\sin(u(t)) + w,\ t \in [0,T] \\
&\quad x(0) = 0,\ y(0) = 1,\ x(T) = 0,\ y(T) = 0 \\
&\quad |u(t)| \le \pi/2,\ t \in [0,T]\ .
\end{aligned}$$

For $V = 1$, $w = 1/\sqrt{2}$ and using the initial guesses $x(t_i) = 0, y(t_i) = 1, i \in I_N$ and $u(t_i) = 0, i \in I$, the results are shown in Table 2 for $N = 11$. A value near $T = 3.5$ is exhibited in [9]. The optimal states $x$, $y$ and control $u$ (from the run based on the change of variable $T \to \tau$) are shown in Figs. 1(g) and 1(h) respectively. Figure 1(i) presents the optimal control obtained from the run that adds a new variable to the states vector.

The next three problems are OCP of the *Lagrange form* and the last problem is already in the *Mayer form*.

**Problem 4** In a continuous stirred-tank chemical reactor (*Tank reactor*), $y_1$ represents the deviation from the steady-state temperature, $y_2$ represents the deviation from the steady-state concentration and $u$ is the effect of the coolant flow on the chemical reaction [10]:

$$\begin{aligned}
&\min_{u(t)} J \equiv \int_0^T (y_1(t)^2 + y_2(t)^2 + Ru(t)^2)\,dt \\
&\text{s.t. } y_1'(t) = -2(y_1(t) + 0.25) + (y_2(t) + 0.5)\exp\left(\tfrac{25y_1(t)}{y_1(t)+2}\right) \\
&\quad -(y_1(t) + 0.25)u(t) \\
&\quad y_2'(t) = 0.5 - y_2(t) - (y_2(t) + 0.5)\exp\left(\tfrac{25y_1(t)}{y_1(t)+2}\right),\ t \in [0,T] \\
&\quad y_1(0) = 0.05,\ y_2(0) = 0\ .
\end{aligned}$$

The optimal solution reported in [10], for $T = 0.78$ and $R = 0.1$, is $J^* = 0.0268$. Using the initial guesses $y_1(t_i) = 0.05$, $y_2(t_i) = 0, i \in I_N$ and $u(t_i) = 0.75, i \in I$, with $N = 11$, the results are shown in Table 3. The proposed strategy has produced again a reasonably good solution. Figures 2(a) and 2(b) show the optimal states $y_1, y_2$ and control $u$ respectively.

**Problem 5** In the point mass maximum travel example (*masstravel*), the force $u(t)$ that moves a mass to the longest distance is to be found (with $T = 10$ fixed):

$$\begin{aligned}
&\max_{u(t)} J \equiv \int_0^T v(t)\,dt \\
&\text{s.t. } s'(t) = v(t) \\
&\quad v'(t) = u(t) - k_0 - k_1 v(t) - k_2 v(t)^2,\ t \in [0,T] \\
&\quad s(0) = 0,\ v(0) = 0,\ v(T) = 0 \\
&\quad |u(t)| \le g + k_3 v(t)^2,\ t \in [0,T]\ .
\end{aligned}$$

The results, for $k_0 = 0.1, k_1 = 0.2, k_2 = 1, k_3 = 1$ and $N = 11$, are shown in Table 3. The initial guesses were $s(t_i) = 1$, $v(t_i) = 2, i \in I_N$ and $u(t_i) = 5, i \in I$. When

**Table 3** Results obtained for the problems 4, 5, 6 and 7

| | $k$ | $J$ | $\theta$ | $p$ | $nfe$ | $time$ |
|---|---|---|---|---|---|---|
| *Tank reactor* | 1 | 0.0046 | $1.12E-02$ | $0.0000E+00$ | | |
| | 176 | 0.0357 | $9.9503E-05$ | $0.0000E+00$ | 16320 | 18.0 |
| *masstravel* | 1 | 3.2633 | $6.9821E+01$ | $1.6000E+02$ | | |
| | 69 | 6.0311 | $7.9855E-05$ | $0.0000E+00$ | 4830 | 5.3 |
| | 128[§] | 6.0256 | $9.2528E-11$ | $0.0000E+00$ | 8963 | 9.7 |
| *trajectory* | 1 | 0.6457 | $1.6043E+01$ | $1.0424E+01$ | | |
| | 56 | 0.2691 | $9.3978E-05$ | $0.0000E+00$ | 3922 | 4.4 |
| | 307[§] | 0.2635 | $8.8477E-11$ | $0.0000E+00$ | 21494 | 22.5 |
| *obstacle* | 1 | 0.0000 | $2.4395E+00$ | $0.0000E+00$ | | |
| | 341 | 2.3257 | $9.2300E-05$ | $2.5452E-05$ | 26208 | 27.1 |
| | 750[§] | 2.4616 | $1.3062E-08$ | $4.8821E-10$ | 52702 | 53.7 |

transforming the above form into the *Mayer form*, the objective function value is just $s(T)$ (thus no new state variable was added to the states vector). To confirm convergence, the problem is also solved with $\eta_1 = 1E-10$, $\eta_2 = 1E-06$ in (11) – identified with [§] in Table 3. Figures 2(c) and 2(d) contain the states and control respectively.

**Problem 6** (*trajectory*) Find $u(t)$ that minimizes $J$ (with $T = 3$ fixed) [4],

$$\begin{aligned}
&\min_{u(t)} J \equiv \int_0^T (y^2(t) + u^2(t))\, dt \\
&\text{s.t. } y'(t) = (1 + y(t))y(t) + u(t),\ t \in [0,T] \\
&\quad y(0) = 0.05,\ y(T) = 0, \\
&\quad |y(t)| \le 1,\ |u(t)| \le 1,\ t \in [0,T]\ .
\end{aligned}$$

The obtained results for $N = 11$, with the initial guesses $y(t_i) = 1, i \in I_N$ and $u(t_i) = 0, i \in I$, are displayed in Table 3. Results with $\eta_1 = 1E-10$, $\eta_2 = 1E-06$ in (11) are also included. The Figs. 2(e) and 2(f) present the states and control respectively.

**Problem 7** The obstacle problem (*obstacle*) can be reformulated as [3] ($T = 2.9$):

$$\begin{aligned}
&\min_{u(t)} J \equiv 5y_1(T)^2 + y_2(T)^2 \\
&\text{s.t. } y_1'(t) = y_2(t) \\
&\quad y_2'(t) = u(t) - 0.1(1 + 2y_1(t)^2)y_2(t) \\
&\quad y_1(0) = 1,\ y_2(0) = 1, \\
&\quad 1 - 9(y_1(t) - 1)^2 - (\tfrac{y_2(t) - 0.4}{0.3})^2 \le 0, \\
&\quad -0.8 - y_2(t) \le 0,\ |u(t)| \le 1,\ t \in [0,T]
\end{aligned}$$

Using the initial guesses $y_1(t_i) = 0, y_2(t_i) = 0, i \in I_N$, $u(t_i) = 0, i \in I$ and $N = 11$, the results are shown in Table 3. This problem is also solved with $\eta_1 = 1E-10$, $\eta_2 = 1E-06$ in (11) to analyze the convergence issue. Figures 2(g) and 2(h) show the states $y_1, y_2$ and control $u$ respectively.

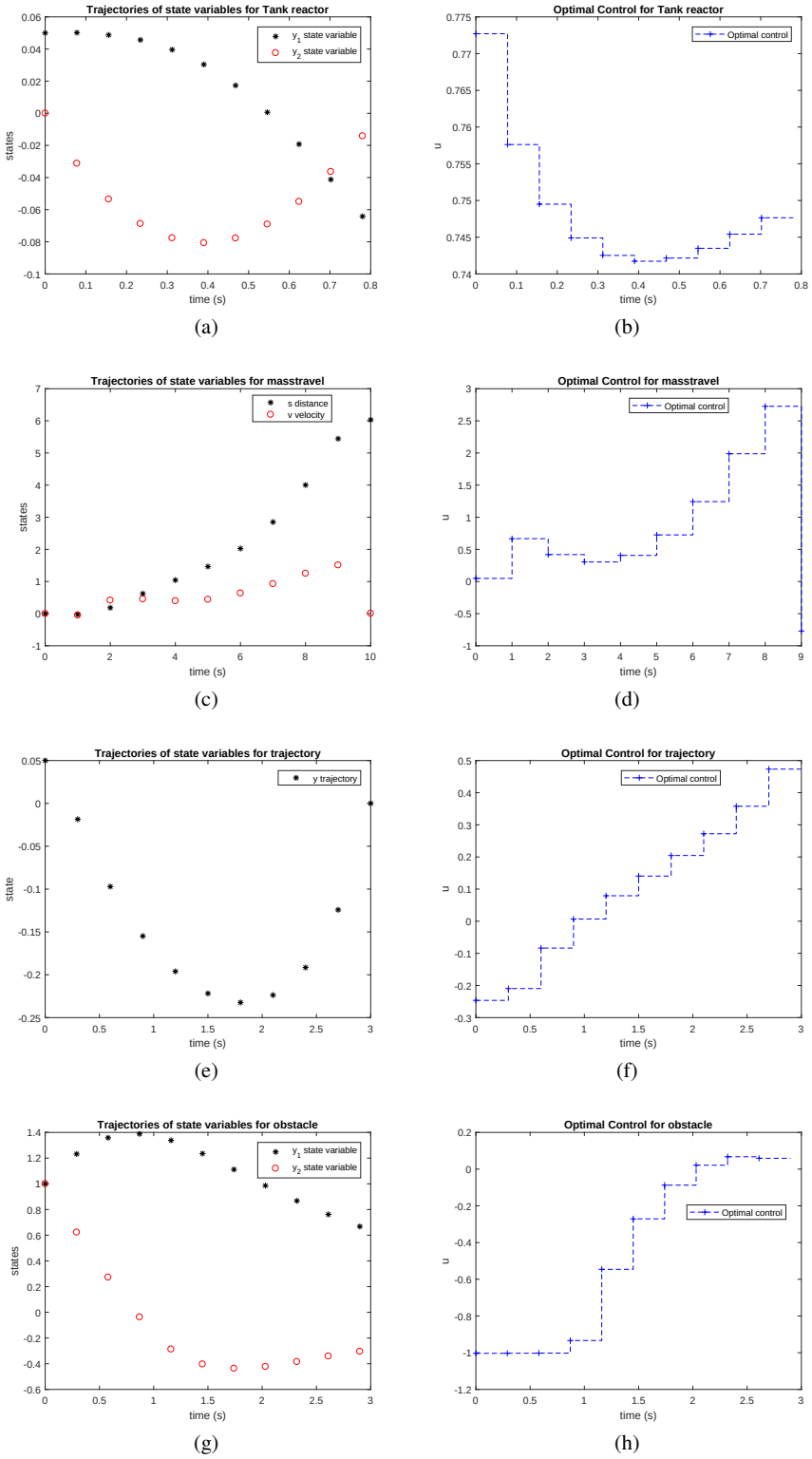


**Fig. 2** States trajectory and optimal control. **a** States for problem *Tank reactor*. **b** Control for problem *Tank reactor*. **c** States for problem *masstravel*. **d** Control for problem *masstravel*. **e** States for problem *trajectory*. **f** Control for problem *trajectory*. **g** States for problem *obstacle*. **h** Control for problem *obstacle*.

## 5 Conclusions

A first-order descent method based on a filter methodology is proposed to solve a finite-dimensional nonlinear optimization problem that arises from the use of a direct multiple shooting method for OCP. The implemented filter method relies on three measures. The two feasibility measures are handled separately in order to give priority to the minimization of the 'continuity constraints' violation over the algebraic equality and inequality constraints violation and the objective function. This priority is patent by the use of search directions that are along either the negative of the gradient of the 'continuity constraints' violation function or a negative convex combination of that gradient and the gradient of the other constraints violation, or the objective function. Numerical derivatives are implemented in order to avoid computing the first derivatives of the involved functions. The numerical experiments carried out until now have shown that the presented strategy is worth pursuing.

Issues related to the extension of the proposed method to solving retarded OCP with constant delays in the state variables and in the control are now under investigation and will be the subject of a future paper.

**Acknowledgements** We acknowledge the financial support of CIDEM, R&D unit funded by the FCT - Portuguese Foundation for the Development of Science and Technology, Ministry of Science, Technology and Higher Education, under the Project UID/EMS/0615/2019, and FCT within the Projects Scope: UID/CEC/00319/2019 and UID/MAT/00013/2013.

## References

1. Biegler L (2010) Nonlinear programming: concepts, algorithms, and applications to chemical processes. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA
2. Frego M (2014) Numerical Methods for Optimal Control Problems with Applications to Autonomous Vehicles. Ph.D. Thesis, University of Trento
3. Schlegel M, Stockmann K, Binder T et al (2005) Dynamic optimization using adaptive control vector parameterization. Comput Chem Eng 29(8):1731–1751
4. Diehl M, Bock HG, Diedam H et al (2006) Fast direct multiple shooting algorithms for optimal robot control. In: Diehl M, Mombaur K (eds) Fast motions in biomechanics and robotics, Lecture Notes in Control and Information Sciences 340:65–93
5. Fletcher R, Leyffer S (2002) Nonlinear programming without a penalty function. Math Program Series A 91(2):239–269
6. Audet C, Dennis JE Jr (2004) A pattern search filter method for nonlinear programming without derivatives. SIAM J Optimiz 14(4):980–1010
7. Assassa F, Marquardt W (2014) Dynamic optimization using adaptive direct multiple shooting. Comput Chem Eng 60:242–259
8. Schäder A, Kühl P, Diehl M et al (2007) Fast reduced multiple shooting methods for nonlinear model predictive control. Chem Eng Process 46(11):1200–1214
9. How JP 16.323 Principles of Optimal Control. Lecture 7 Numerical Solution in Matlab. Spring 2008. Massachusetts Institute of Technology: MIT OpenCourseWare, `http://ocw.mit.edu`. License: Creative Commons BY-NC-SA.
10. Kirk DE (1970) Optimal control theory: an introduction. Prentice Hall, Inc, NJ